# Simulation halbgeordneter Petrinetz-Abläufe

Thomas Freytag, Universität Karlsruhe, Institut AIFB
tfr@aifb.uni-karlsruhe.de

## 1 Ausgangssituation

Im Rahmen des von der DFG geförderte Projekts **Verifikation von Informationssystemen durch Auswertung halbgeordneter Petrinetz-Abläufe (VIP)** (vgl. [DO94], [DO95]) wird in Kooperation zwischen den Universitäten Karlsruhe und Frankfurt versucht, das Effizienzpotential des halbgeordneten Ansatzes bei der Simulation von Petrinetzen nutzbar zu machen. Ziele sind folgende:

- Modellierung verteilter Informationssysteme mit höheren Petrinetzen
- Simulation auf der Basis halbgeordneter Abläufe (Prozeßnetze)
- Entwurf einer Anfragesprache zur Formulierung von Systemeigenschaften
- Prototypische Umsetzung der Konzepte in ein Software-Werkzeug
- Effiziente Abspeicherung und Verwaltung von Prozeßnetzen
- Entwurf einer benutzerfreundlichen Schnittstelle
- Einbettung in die Werkzeugumgebung INCOME ([OSS94])

Im diesem Beitrag soll anhand eines typischen Beispiels der Effizienzvorteil bei der Verifikation von Systemeigenschaften durch Simulation halbgeordneter Abläufe gegenüber dem "klassischen" Schaltfolgen-Ansatz angedeutet und der aktuelle Stand des Projekts vorgestellt werden.

## 2 Grundlegende Konzepte

### 2.1 Betrachtete Netzklasse

Es wird von Pr/T-Netzen mit folgenden Einschränkungen ausgegangen:

- Endlicher Netzgraph
- Endliche Domains für Markierungs- und Kantenanschrifts-Tupel
- Keine Mehrfach-Tupel in Markierungen
- Transitionen haben eine nichtleeren Vor- und Nachbereich

Diese Einschränkungen sind hinnehmbar, da das Interesse ohnehin auf die Modellierung und Simulation **realer Systeme** gerichtet ist. Auf eine formale Netzdefinition wird hier verzichtet. Es sei auf die Arbeiten [KW95] und [Jen92] verwiesen.

## 2.2 Prozesse von Pr/T-Netzen

Prozesse (genauer: Prozeßnetze) sind Petrinetze (EN-Systeme) mit folgenden zusätzlichen Eigenschaften:

- Keine Zyklen
- Keine vorwärts- oder rückwärtsverzweigten Stellen
- Transitionen haben einen nichtleeren Vor- und Nachbereich
- Es existiert ein Netzhomomorphismus zum zugrundeliegenden Pr/T-Netz

Die Generierung von Prozessen für ein Pr/T-Netz mit einer gegebenen Startmarkierung geschieht ganz analog zu der für S/T-Netze:

- **Stellen** entsprechen dem Vorhandensein von Markierungen im Pr/T-Netz: $P(1,2)$ ist eine Stelle des Prozeßnetzes, wenn die Stelle $P$ des Pr/T-Netzes mit dem Tupel $\langle 1,2 \rangle$ markiert ist
- **Transitionen** entsprechen Schaltenvorgängen im Pr/T-Netz: $T[x \to 1, y \to 2]$ ist eine Transition des Prozeßnetzes, wenn die Transition $T$ des Pr/T-Netzes für diese Belegung schaltet
- **Kanten** entsprechen dem Markenfluß: Eine Kante von einer Stelle zu einer Transition besagt, daß das Schalten dieser Transition im Pr/T-Netz an den entsprechenden Stellen die jeweiligen Marken verbraucht bzw. erzeugt
- Die **Anfangsmarkierung** besteht aus den Stellen ohne Vorbereich (und nur aus diesen)

# 3 Das Beispiel Aufzugsteuerung

## 3.1 Darstellung als Petrinetz

In Abb. 1 ist ein Aufzugsystem modelliert, bei dem in jedem Stockwerk ein Schalter zum Anfordern eines Aufzuges vorhanden ist. Es gibt zwei stehende Aufzüge ($a$ im Stockwerk 1, $b$ im Stockwerk 5), ausgedrückt durch zwei Tupel auf der Stelle $Steht$. Außerdem gibt es zwei Anforderungen in den Stockwerken 2 und 3, was durch die beiden Tupel auf der Stelle $Anford$ modelliert ist. Das weitere Verhalten des Systems hängt nun davon ab, welcher Aufzug zur Befriedigung welcher Anforderung eingesetzt wird. Eine Fakt-Transition (vgl. [GTM76]) $Auf\_und\_ab$ soll hier die Forderung ausdrücken, daß ein Aufzug nie gleichzeitig in der Aufwärts- und Abwärtsbewegung sein kann.

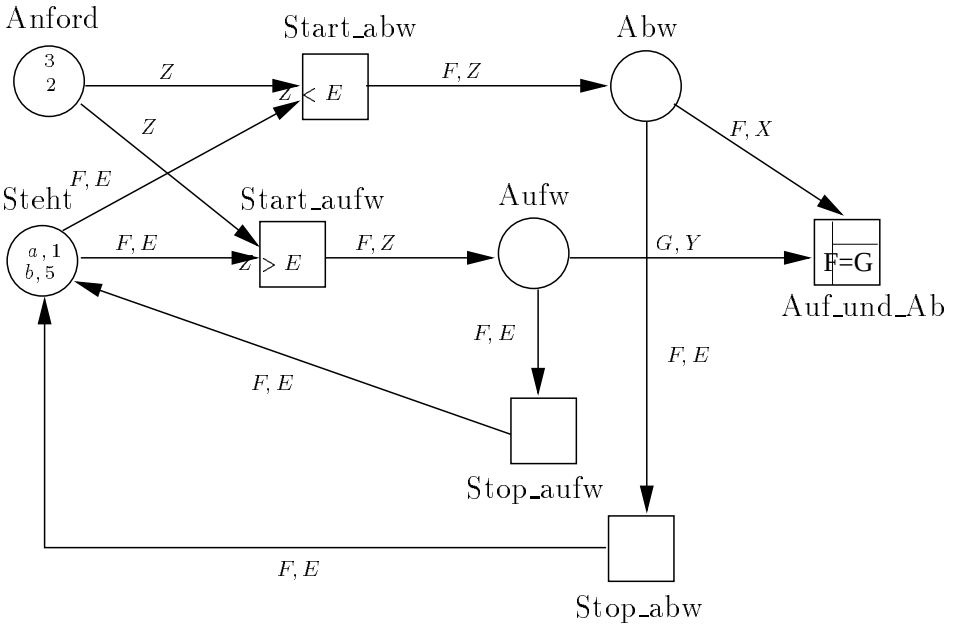


Abbildung 1: *Ein "Informationssystem" als Pr/T-Netz (Aufzugsteuerung)*

## 3.2 Schaltfolgen und erreichbare Markierungen

Das Beispielnetz aus Abb. 1 hat eine Erreichbarkeitsmenge mit 25 Markierungen, wie Tab. 1 zeigt. Um die durch den Fakt $Auf\_und\_ab$ spezifizierte Netzeigenschaft durch Simulation von Schaltfolgen zu verifizieren, muß der gesamte Zustandsraum (d.h. alle erreichbaren Markierungen) konstruiert und daraufhin durchsucht werden, ob die Transition aktiviert ist oder nicht. Dies erfordert im allgemeinen Fall exponentiellen Aufwand in Abhängigkeit von der Anzahl der Stellen und Marken.

## 3.3 Prozesse

Bei einer Simulation von halbgeordneten Abläufen ergibt sich eine weitaus kompaktere Darstellung der Resultate (Abb. 2). Es gibt insgesamt sechs (maximale) Prozesse für das Beispielnetz:

- Die Prozeßnetze $P_1$ und $P_2$ repräsentieren die Konfliktauflösung, welcher der beiden Aufzüge welche Anforderung (jeweils nebenläufig) bedient
- Die Prozeßnetze $P_3$ bis $P_6$ stellen die Fälle dar, in denen ein Aufzug beide Anforderungen bedient und der jeweils andere nicht fahrbereit ist

Zur Verifikation des Fakts $Auf\_und\_ab$ reicht es aus, zu zeigen, daß in keinem der von der Simulation erzeugten Prozesse zwei Ereignisse der Form $Aufw(x, i_1)$ und $Abw(x, i_2)$ ungeordnet vorkommen. Diese Suche ist im allgemeinen mit quadratischem (bei optimiertem Algorithmus evtl. sogar besserem) Aufwand möglich.

| **Mrk** | **Anford** | **Steht** | **Abw** | **Aufw** | **[Belegung ⟩ Folge-Mark.** |
|---|---|---|---|---|---|
| $M_0$ | $< 2 >, < 3 >$ | $< a, 1 > < b, 5 >$ | | | [Start_aufw(F=a,E=1,Z=2)⟩$M_1$<br>[Start_aufw(F=a,E=1,Z=3)⟩$M_2$<br>[Start_abw(F=b,E=5,Z=2)⟩$M_3$<br>[Start_abw(F=b,E=5,Z=3)⟩$M_4$ |
| $M_1$ | $< 3 >$ | $< b, 5 >$ | | $< a, 2 >$ | [Stop_aufw(F=a,E=2)⟩$M_5$<br>[Start_abw(F=b,E=5,Z=3)⟩$M_8$ |
| $M_2$ | $< 2 >$ | $< b, 5 >$ | | $< a, 3 >$ | [Stop_aufw(F=a,E=3)⟩$M_6$<br>[Start_abw(F=b,E=5,Z=2)⟩$M_7$ |
| $M_3$ | $< 3 >$ | $< a, 1 >$ | $< b, 2 >$ | | [Start_aufw(F=a,E=1,Z=3)⟩$M_7$<br>[Stop_abw(F=b,E=2)⟩$M_9$ |
| $M_4$ | $< 2 >$ | $< a, 1 >$ | $< b, 3 >$ | | [Start_aufw(F=a,E=1,Z=2)⟩$M_8$<br>[Stop_abw(F=b,E=3)⟩$M_{10}$ |
| $M_5$ | $< 3 >$ | $< a, 2 >, < b, 5 >$ | | | [Start_abw(F=a,E=2,Z=3)⟩$M_{11}$<br>[Start_aufw(F=b,E=5,Z=3)⟩$M_{13}$ |
| $M_6$ | $< 2 >$ | $< a, 3 >, < b, 5 >$ | | | [Start_abw(F=a,E=2,Z=3)⟩$M_{12}$<br>[Start_abw(F=b,E=5,Z=2)⟩$M_{14}$ |
| $M_7$ | | | $< b, 2 >$ | $< a, 3 >$ | [Stop_aufw(F=a,E=3)⟩$M_{14}$<br>[Stop_abw(F=b,E=2)⟩$M_{15}$ |
| $M_8$ | | | $< b, 3 >$ | $< a, 2 >$ | [Stop_aufw(F=a,E=2)⟩$M_{13}$<br>[Stop_abw(F=b,E=3)⟩$M_{16}$ |
| $M_9$ | $< 3 >$ | $< a, 1 >, < b, 2 >$ | | | [Start_aufw(F=a,E=1,Z=3)⟩$M_{15}$<br>[Start_aufw(F=b,E=2,Z=3)⟩$M_{17}$ |
| $M_{10}$ | $< 2 >$ | $< a, 1 >, < b, 3 >$ | | | [Start_aufw(F=a,E=1,Z=2)⟩$M_{16}$<br>[Start_abw(F=b,E=3,Z=3)⟩$M_{18}$ |
| $M_{11}$ | | $< b, 5 >$ | | $< a, 3 >$ | [Stop_aufw(F=a,E=3)⟩$M_{19}$ |
| $M_{12}$ | | $< b, 5 >$ | | $< a, 2 >$ | [Stop_abw(F=a,E=2)⟩$M_{20}$ |
| $M_{13}$ | | $< a, 2 >$ | | $< b, 3 >$ | [Stop_abw(F=b,E=3)⟩$M_{21}$ |
| $M_{14}$ | | $< a, 3 >$ | | $< b, 2 >$ | [Stop_abw(F=b,E=2)⟩$M_{22}$ |
| $M_{15}$ | | $< b, 2 >$ | | $< a, 3 >$ | [Stop_aufw(F=a,E=3)⟩$M_{22}$ |
| $M_{16}$ | | $< b, 3 >$ | | $< a, 2 >$ | [Stop_aufw(F=a,E=2)⟩$M_{21}$ |
| $M_{17}$ | | $< a, 1 >$ | $< b, 3 >$ | | [Stop_aufw(F=b,E=3)⟩$M_{23}$ |
| $M_{18}$ | | $< a, 1 >$ | $< b, 2 >$ | | [Stop_abw(F=b,E=2)⟩$M_{24}$ |
| $M_{19}$ | | $< a, 3 > < b, 5 >$ | | | |
| $M_{20}$ | | $< a, 2 > < b, 5 >$ | | | |
| $M_{21}$ | | $< a, 2 > < b, 3 >$ | | | |
| $M_{22}$ | | $< a, 3 > < b, 2 >$ | | | |
| $M_{23}$ | | $< a, 1 > < b, 3 >$ | | | |
| $M_{24}$ | | $< a, 1 > < b, 2 >$ | | | |

Tabelle 1: Erreichbarkeitstabelle für das Netz in Abb. 1

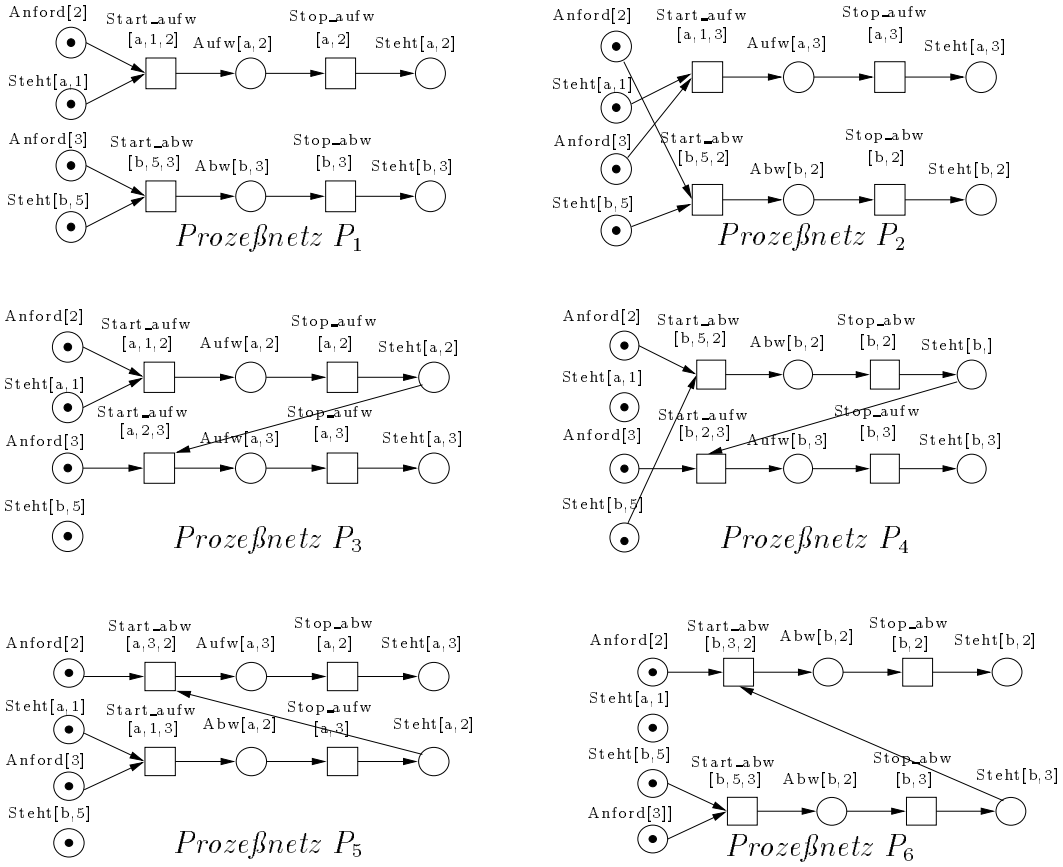


Abbildung 2: *Prozeßnetze für das Beispiel aus Abb. 1*

# 4 Auswertung der Abläufe

Das einfache Gegenüberstellen der Schaltfolgen und der Prozesse des Beispiels deutet an, wie vielversprechend der Ansatz der halbordnungsbasierten Simulation in Bezug auf Effizienzverbesserung ist. Die Lösung der folgenden Fragestellungen wird dabei im Vordergrund stehen:

- Wo werden unendliche Prozesse (die ja bei lebendigen Pr/T-Netzen stets vorliegen) "abgeschnitten", ohne daß für die Verifikation benötigte Informationen verloren gehen? Hier gibt es in der Literatur ([Esp94], [McM92]) eine Reihe von Ansätzen im Rahmen von Entfaltungen von S/T-Systemen, die auf Prozesse von Pr/T-Netze übertragen werden müssen.
- Wie kann die Prozeßgenerierung von außen (abhängig von der zu verifizierenden Eigenschaft) so gesteuert werden, daß möglichst nur diejenigen Prozesse konstruiert werden, die zur Problemlösung gebraucht werden?
- Wie können Prozeßnetze effizient abgespeichert und graphisch visualisiert werden?

# 5 Software-Konzept

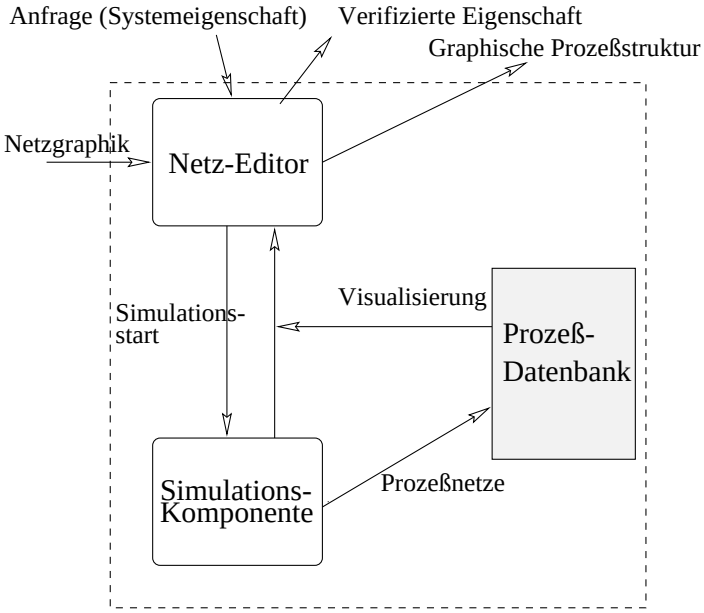


Abbildung 3: *Die Software-Struktur des VIP-Werkzeugs*

Geplant ist das in Abb. 3 dargestellte folgende Simulationskonzept:

- Der Benutzer erstellt ein Netzmodell und startet die Simulation .Der Simulator generiert - gesteuert durch einen intelligenten Algorithmus zur Bestimmung der Terminierungsbedingung - halbgeordnete Abläufe und speichert diese in geeigneter Form in einer Prozeßdatenbank ab
- Alternativ dazu kann der Benutzer auch im Editor eine graphische Anfrage (zu verifizierende Systemeigenschaft) formulieren. In diesem Fall soll der Simulator in der Lage sein, "selektiv" diejenigen Prozesse zu generieren, die für die Verifikation relevant sind
- In beiden Fällen kann der Benutzer nach Wunsch Abläufe durch die Anzeige von Prozeßnetzen visualisieren

Die beiden Hauptkomponenten von **VIP** sind ein graphischer Netzeditor und eine auf Halbordnungen aufbauende Simulationskomponente. Beim Netzeditor (vgl. Abb. 4) handelt es sich um ein SMALLTALK-Programm, daß mit einem in PROLOG geschriebenen Simulator gekoppelt ist. Die verwendete VIP-Netzbeschreibungssprache (Details in [Fre96]) beruht auf der PROLOG-Syntax und kann direkt vom PROLOG-Interpreter eingelesen und ausgeführt werden. Die Software befindet sich derzeit noch in einem frühen Entwicklungsstadium.

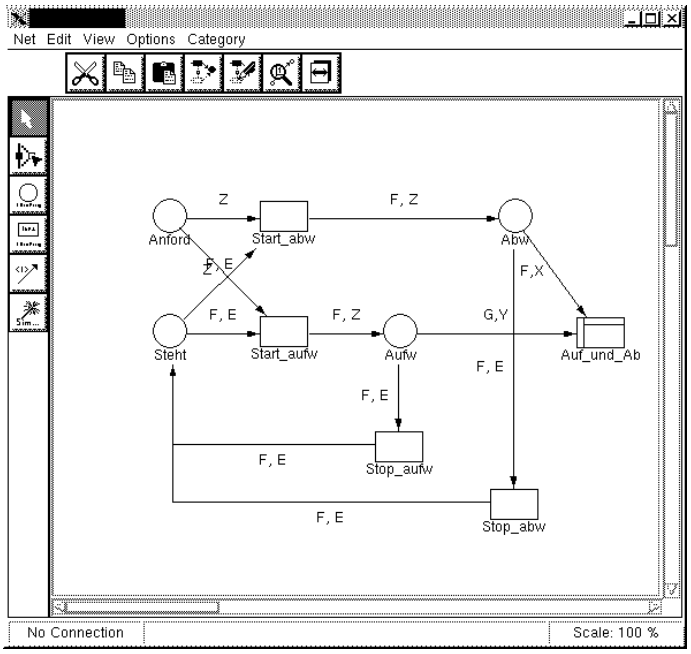


Abbildung 4: *Der VIP-Netzeditor*

# Literatur


[DO94] J. Desel und A. Oberweis. Validierung von Informationssystemen durch Auswertung halbgeordneter Petrinetz-Simulationsläufe. Seiten 132–139. Uni Hannover, Informatik-Berichte Nr. 3, 1994.

[DO95] J. Desel und A. Oberweis. Verifikation von Informationssystemen durch Auswertung halbgeordneter Petrinetz-Abläufe. Forschungsbericht 324, AIFB, Uni Karlsruhe, 1995.

[Esp94] J. Esparza. Model checking using net unfoldings. *Science of Computer Programming*, (23):151–195, 1994.

[Fre96] T. Freytag. Ein Dateiformat für Petrinetze. Interner Projektbericht, AIFB Karlsruhe, Juli 1996.

[GTM76] H. Genrich und G. Thieler-Mevissen. The Calculus of Facts. In A. Mazurkiewicz, Hrsg., *Mathematical Foundations of Computer Science*, Seiten 588–595. Springer-Verlag, 1976.

[Jen92] K. Jensen. *Coloured Petri Nets, Vol.1: Basic Concepts.* Springer-Verlag, Berlin, 1992.

[KW95] E. Kindler und R. Walter. Arc-Typed Petri Nets. Forschungsbericht 50, Humboldt-Universität Berlin, 1995.

[McM92] K.L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *Proceedings of the 4th Workshop on Computer Aided Verification, Montreal*, Seiten 164–174, 1992.

[OSS94] A. Oberweis, V. Sänger und W. Stucky. INCOME/STAR - Methodology and tools for the development of distributed information systems. *Information Systems*, 19(8):641–658, 1994.